

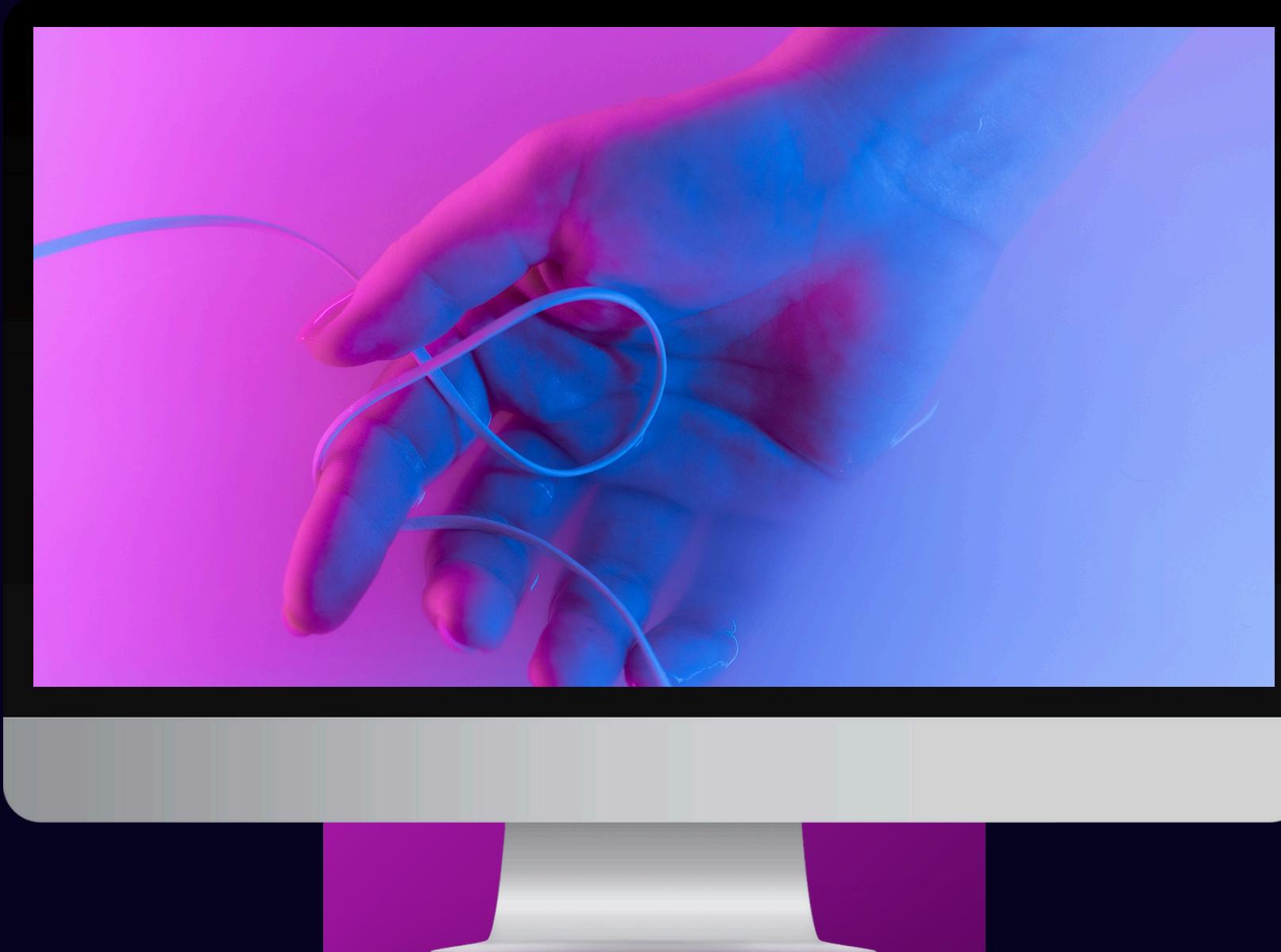


««« ESERCIZIO.PR »»»

ATICO

U3_W2_L5

W W W . R E A L L Y G R E A T S I T E . C O M



TRACCIA:

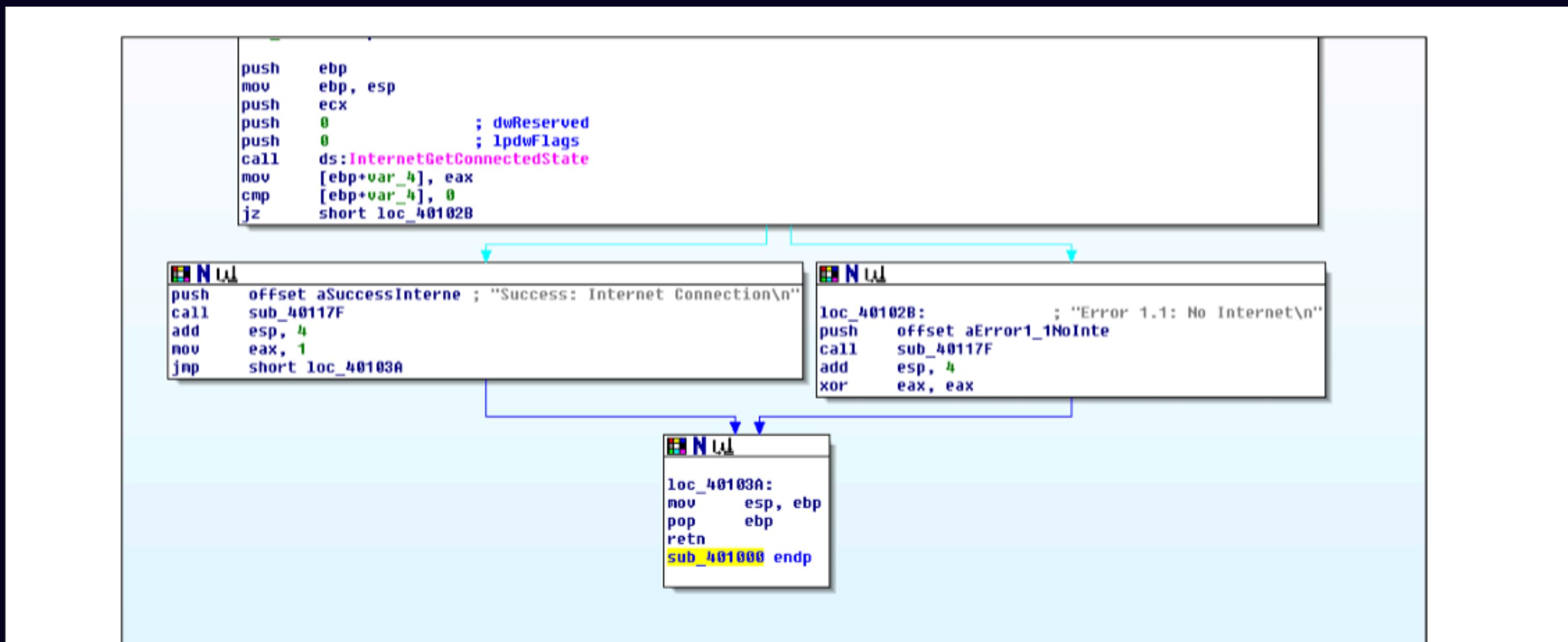
Con riferimento al file Malware_U3_W2_L5 presente all'interno della cartella «Esercizio_Pratico_U3_W2_L5 » sul desktop della macchina virtuale dedicata per l'analisi dei malware, rispondere ai seguenti quesiti:

1. Quali librerie vengono importate dal file eseguibile ?
Fare anche una descrizione

2. Quali sono le sezioni di cui si compone il file eseguibile del malware? Fare anche una descrizione



IMMAGINE SLIDE 3





LE

LIBRERIE



KERNEL32.DLL

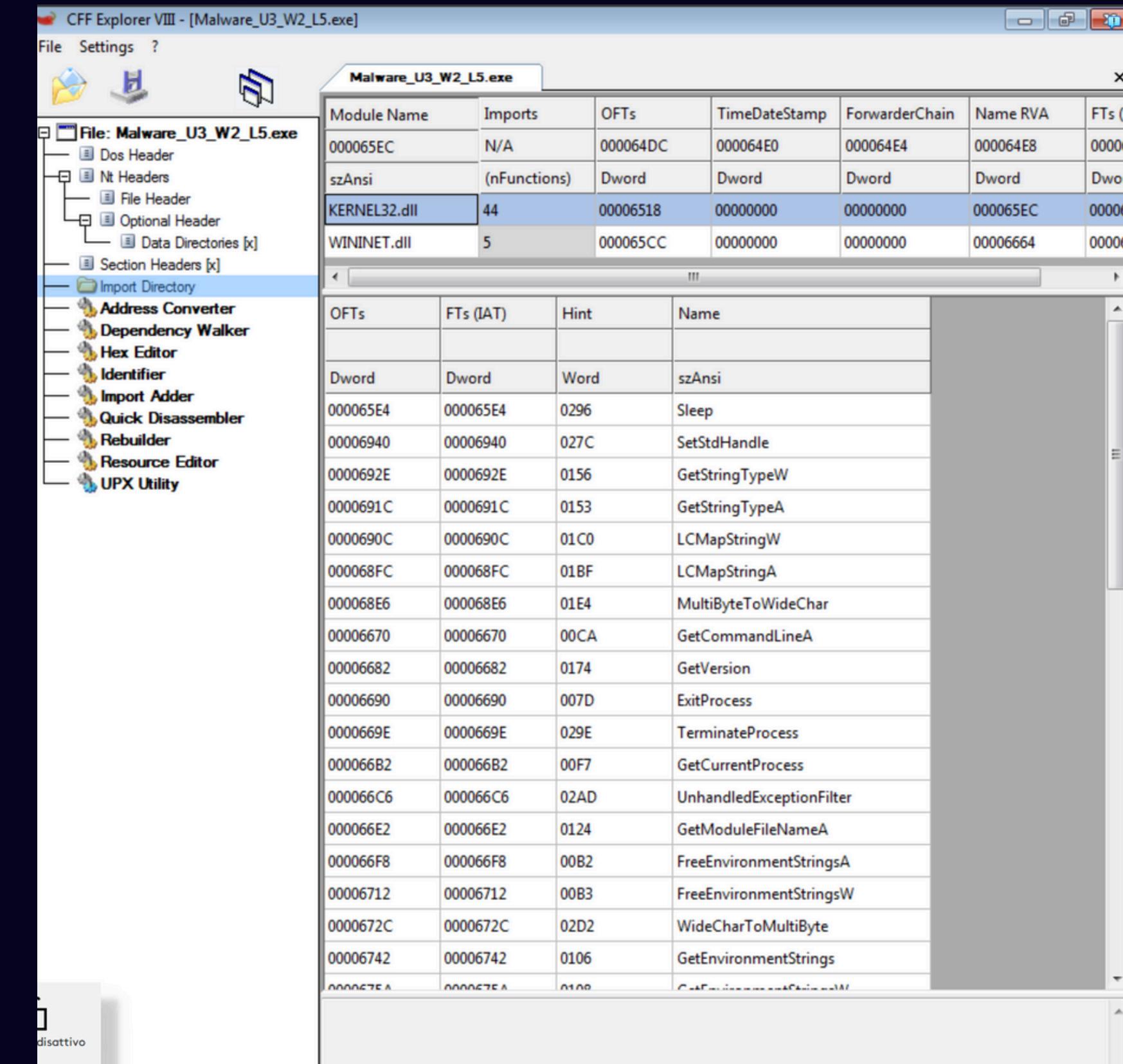
Come possiamo vedere dall'immagine a fianco, la libreria KERNEL.DLL va a richiamare una serie di funzioni che possiamo raggruppare nelle categorie di seguito:

- Gestione dei Processi e del Sistema
- Gestione della Memoria
- Gestione delle Stringhe e delle Variabili d'Ambiente
- Gestione dei Moduli e delle DLL
- Gestione dei File e delle Operazioni I/O
- Gestione degli Errori
- Gestione delle Codifiche e delle Pagine di Codice
- Gestione degli Handle
- Gestione delle Eccezioni e delle Risorse

Queste funzioni importate coprono una vasta gamma di operazioni che il malware può eseguire, dalla gestione della memoria e dei processi alla manipolazione delle stringhe e delle variabili d'ambiente, fino alla gestione delle connessioni Internet e delle DLL.

Comprendere queste funzioni aiuta a capire come il malware interagisce con il sistema operativo, quali misure prende per evitare il rilevamento e come completa le sue attività malevoli.

Questo è fondamentale per sviluppare strategie efficaci di rilevamento e difesa.



WININET.DLL

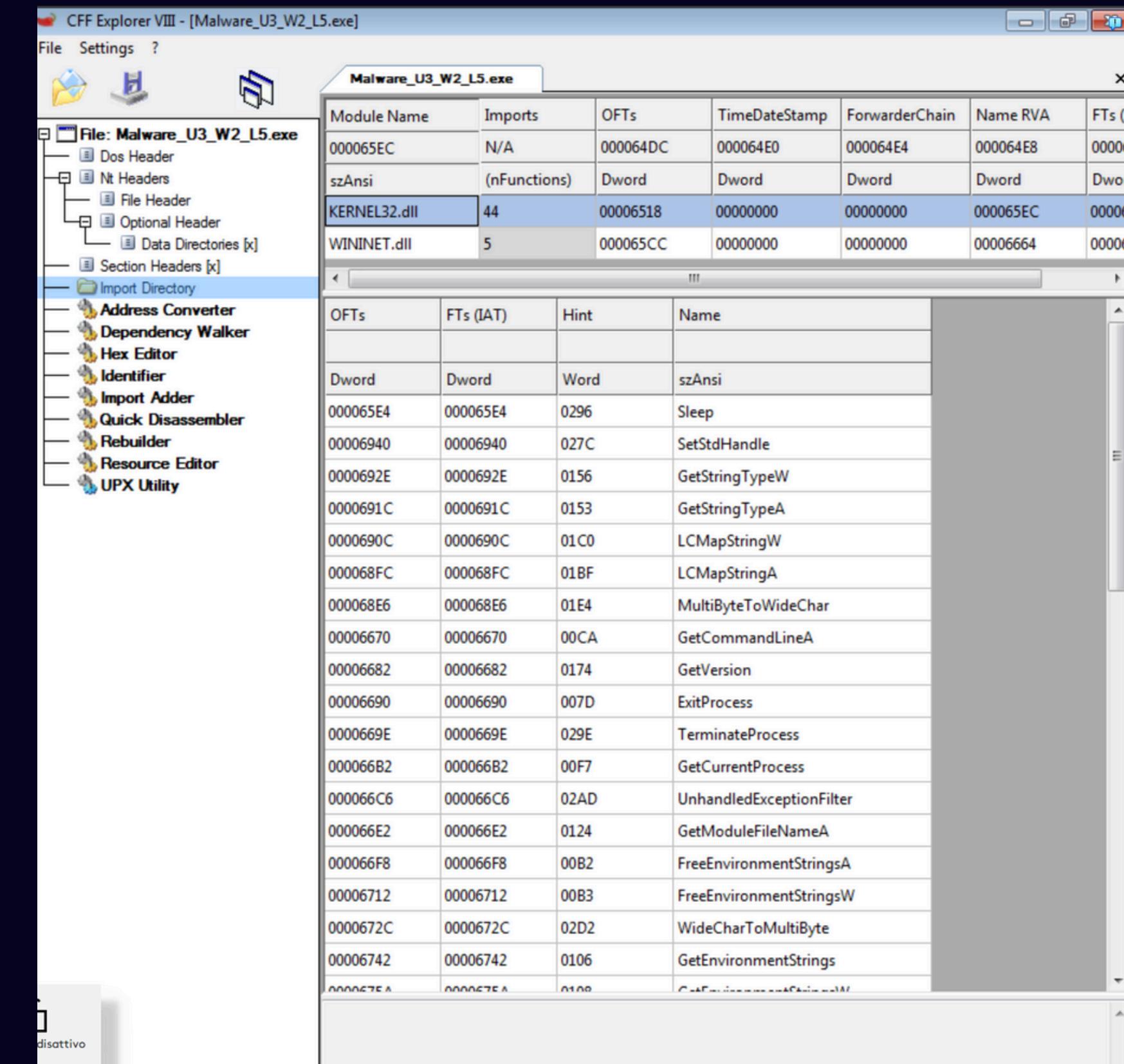
allo stesso modo della libreria kernel.dll, a richiamare una serie di funzioni che possiamo raggruppare nelle categorie di seguito:

- Gestione della Connessione Internet
- Operazioni su URL e Dati
- Accesso a Internet
- Gestione delle Connessioni
- Lettura e Scrittura di Dati

Le funzioni importate dalla DLL WININET.dll dimostrano che il malware è progettato per essere altamente interattivo con Internet, facilitando una comunicazione continua e bidirezionale con server remoti.

Questo aumenta significativamente le sue capacità di download di componenti addizionali, esfiltrazione di dati sensibili e ricezione di comandi per eseguire operazioni specifiche.

La possibilità di verificare lo stato della connessione Internet e gestire le connessioni aperte in modo efficiente rende il malware più adattabile e più difficile da rilevare, complicando gli sforzi di difesa e mitigazione.



Module Name	Imports	OFTs	TimeStamp	ForwarderChain	Name RVA	FTs (I)
000065EC	N/A	000064DC	000064E0	000064E4	000064E8	00006
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dwor
KERNEL32.dll	44	00006518	00000000	00000000	000065EC	00006
WININET.dll	5	000065CC	00000000	00000000	00006664	00006
OFTs	FTs (IAT)	Hint	Name			
Dword	Dword	Word	szAnsi			
000065E4	000065E4	0296	Sleep			
00006940	00006940	027C	SetStdHandle			
0000692E	0000692E	0156	GetStringTypeW			
0000691C	0000691C	0153	GetStringTypeA			
0000690C	0000690C	01C0	LCMapStringW			
000068FC	000068FC	01BF	LCMapStringA			
000068E6	000068E6	01E4	MultiByteToWideChar			
00006670	00006670	00CA	GetCommandLineA			
00006682	00006682	0174	GetVersion			
00006690	00006690	007D	ExitProcess			
0000669E	0000669E	029E	TerminateProcess			
000066B2	000066B2	00F7	GetCurrentProcess			
000066C6	000066C6	02AD	UnhandledExceptionFilter			
000066E2	000066E2	0124	GetModuleFileNameA			
000066F8	000066F8	00B2	FreeEnvironmentStringsA			
00006712	00006712	00B3	FreeEnvironmentStringsW			
0000672C	0000672C	02D2	WideCharToMultiByte			
00006742	00006742	0106	GetEnvironmentStrings			
000067EA	000067EA	0100	GetEnvironmentStringsA			



LE SEZIONI

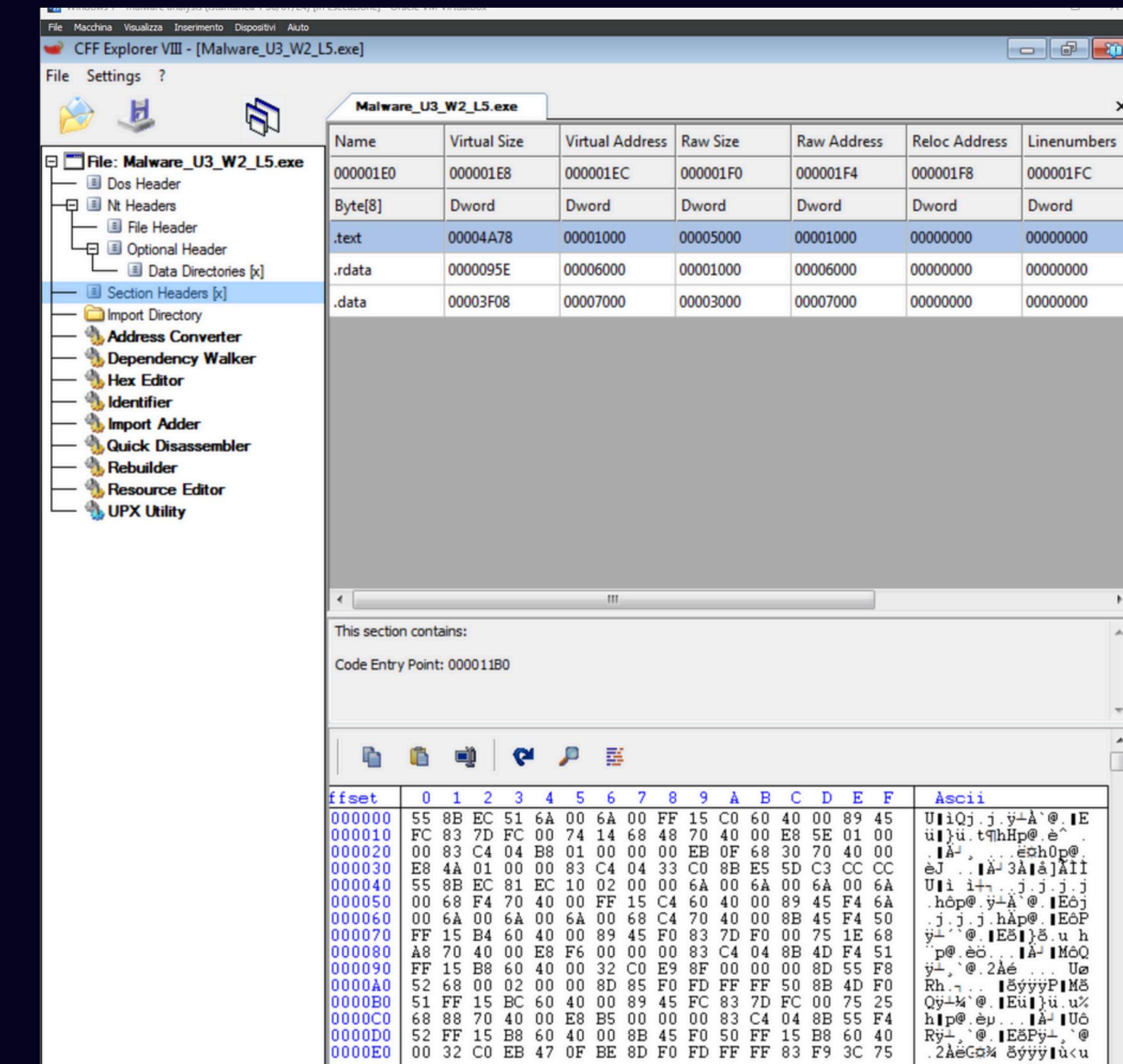




.TEXT

L'immagine mostra il tool CFF Explorer utilizzato per analizzare il file eseguibile di un malware. Le sezioni principali del file eseguibile del malware sono elencate sotto "Section Headers" e includono:

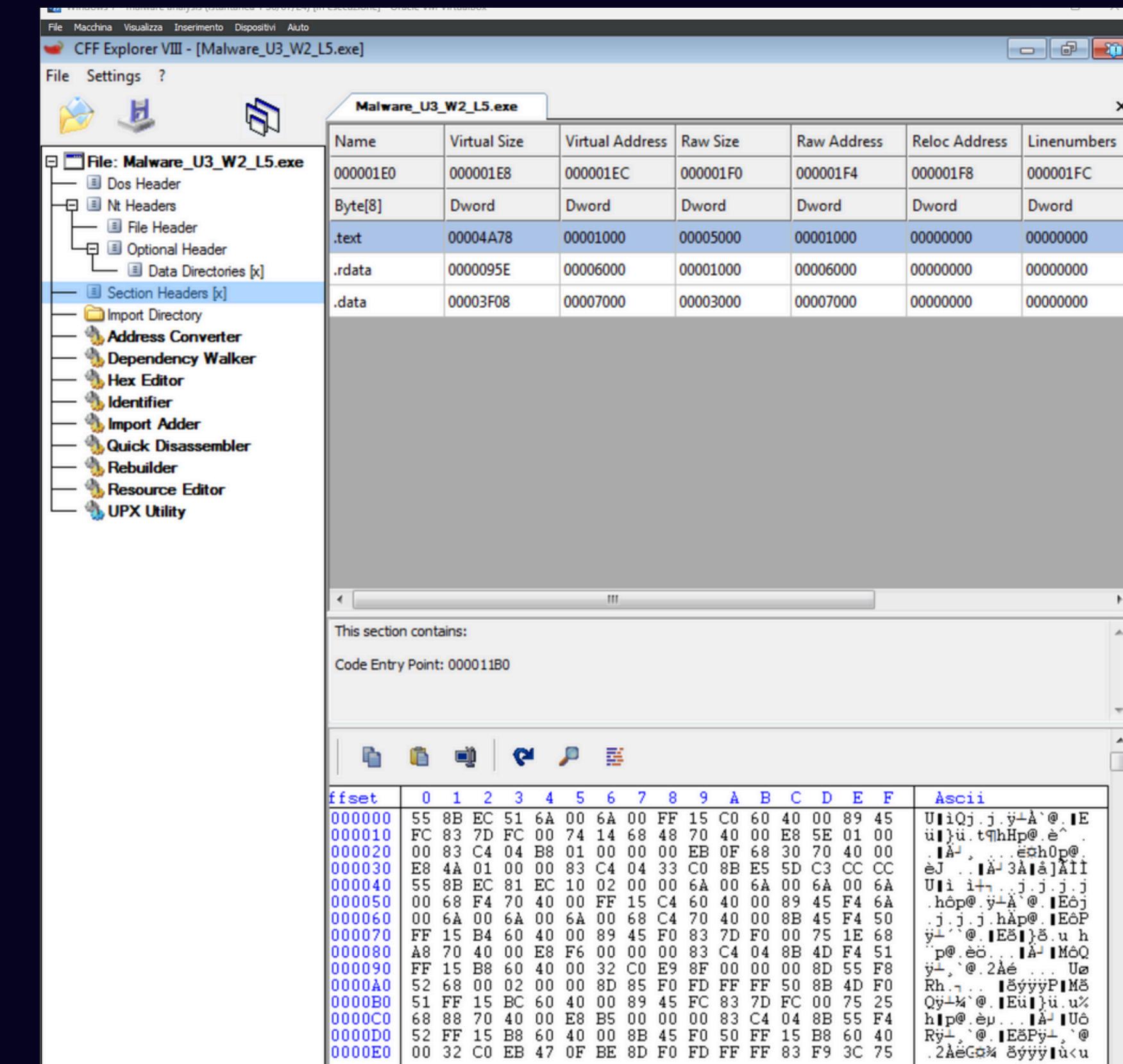
- **Virtual Size:** 00000478
- **Virtual Address:** 00001000
- **Raw Size:** 00000500
- **Raw Address:** 00001000
- **Descrizione:** Questa sezione contiene il codice eseguibile del programma. È la sezione dove risiede il codice macchina del malware e rappresenta la logica principale che il malware eseguirà.



.RDATA

L'immagine mostra il tool CFF Explorer utilizzato per analizzare i file eseguibili di un malware. Le sezioni principali del file eseguibile del malware sono elencate sotto "Section Headers" e includono:

- **Virtual Size:** 0000095E
 - **Virtual Address:** 00006000
 - **Raw Size:** 00000600
 - **Raw Address:** 00006000
 - **Descrizione:** Questa sezione contiene dati di sola lettura, come stringhe, tabelle di funzioni importate e altre informazioni costanti che non vengono modificate durante l'esecuzione del programma.

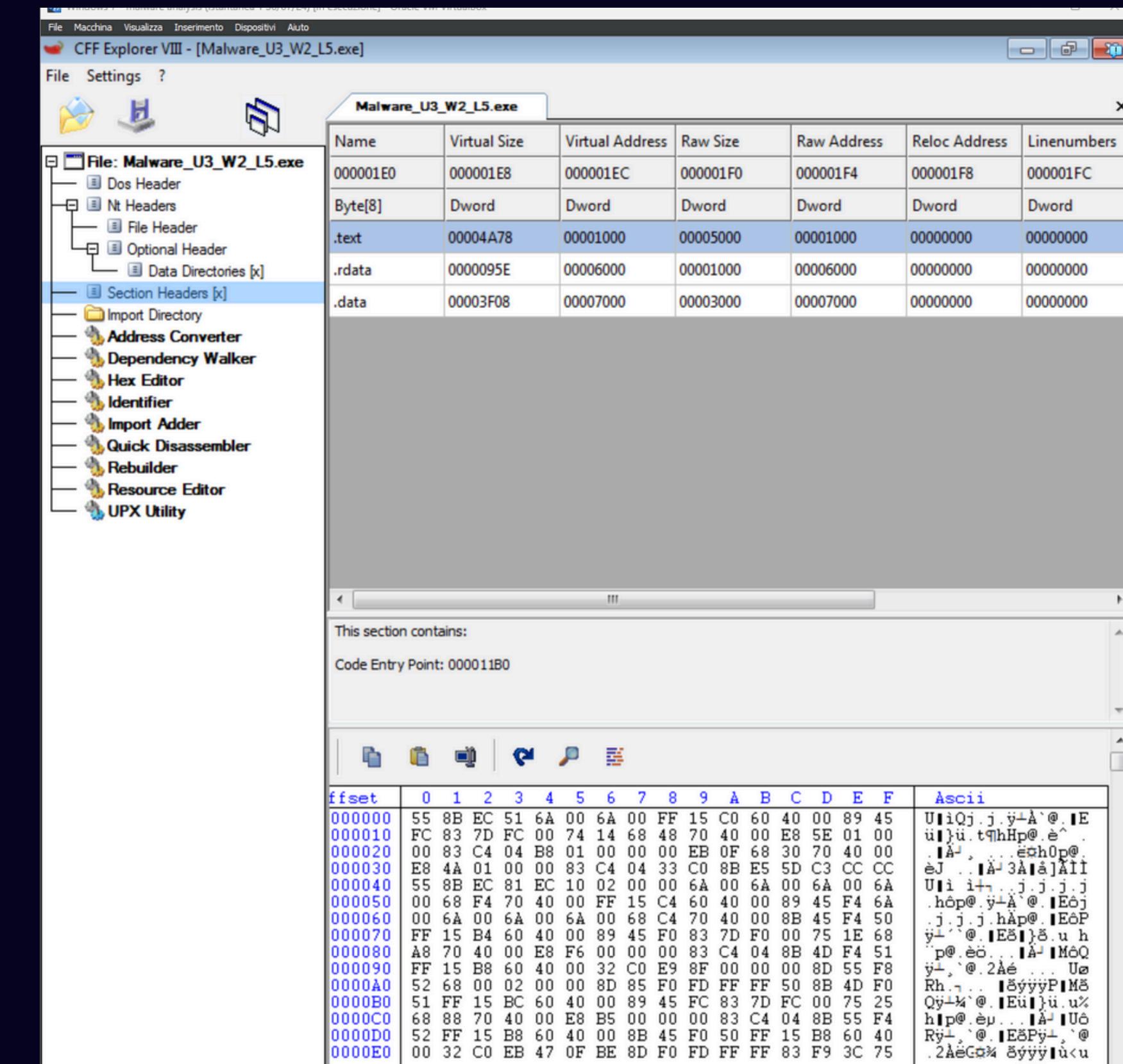




.DATA

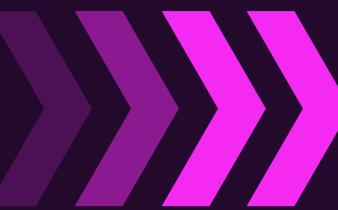
L'immagine mostra il tool CFF Explorer utilizzato per analizzare il file eseguibile di un malware. Le sezioni principali del file eseguibile del malware sono elencate sotto "Section Headers" e includono:

- **Virtual Size:** 0003F08
- **Virtual Address:** 00007000
- **Raw Size:** 0003000
- **Raw Address:** 00007000
- **Descrizione:** Questa sezione contiene dati inizializzati che possono essere letti e scritti durante l'esecuzione del programma. Può includere variabili globali e strutture dati che il programma usa per memorizzare e manipolare informazioni.





SCAN VIRUSTOTAL



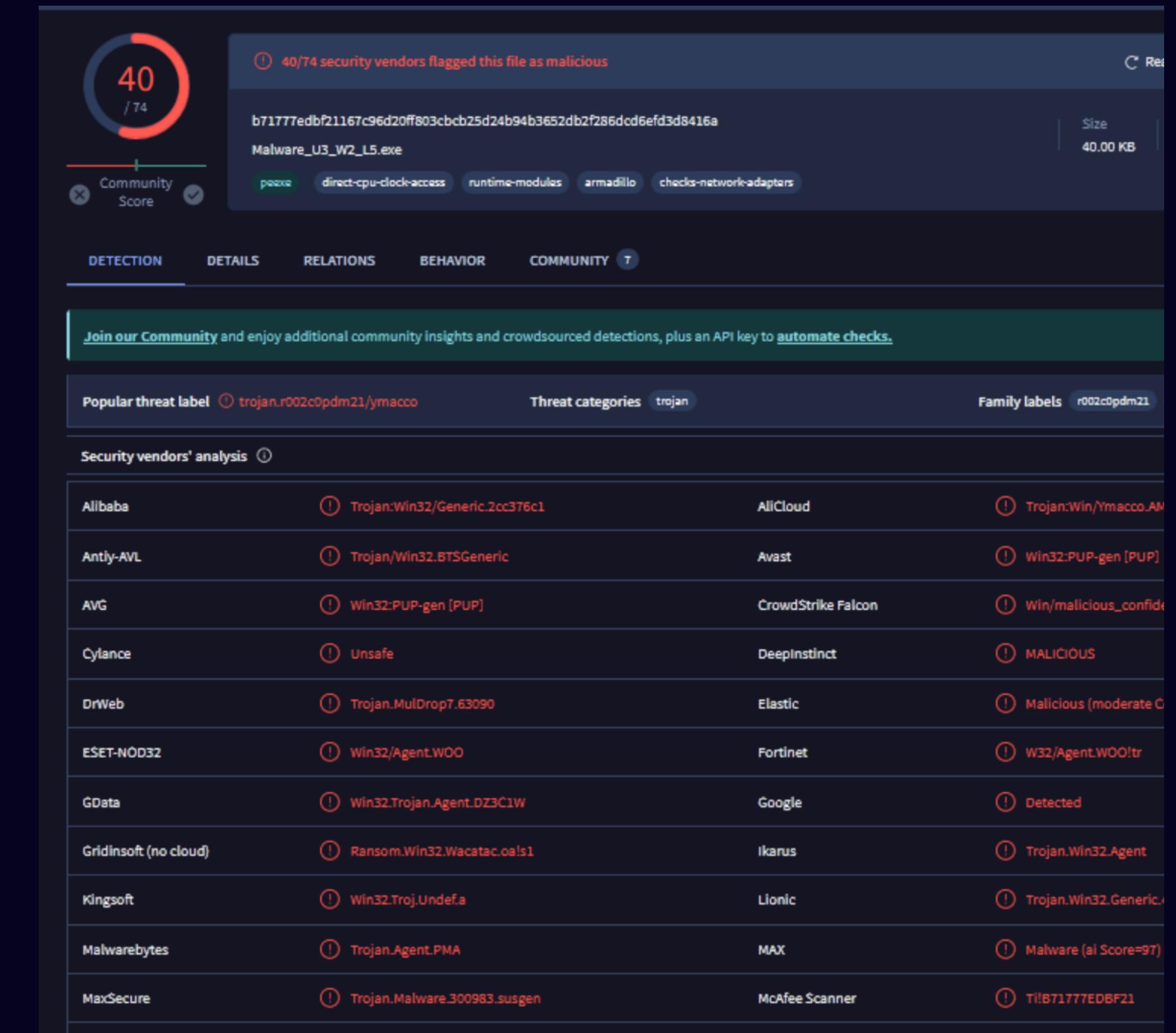


RISULTATO

L'immagine presenta i risultati di un'analisi malware per un file denominato "Malware_U3_W2_L5.exe". Dei 74 motori antivirus utilizzati per l'analisi, 40 hanno rilevato il file come malevolo, classificandolo prevalentemente come un trojan.

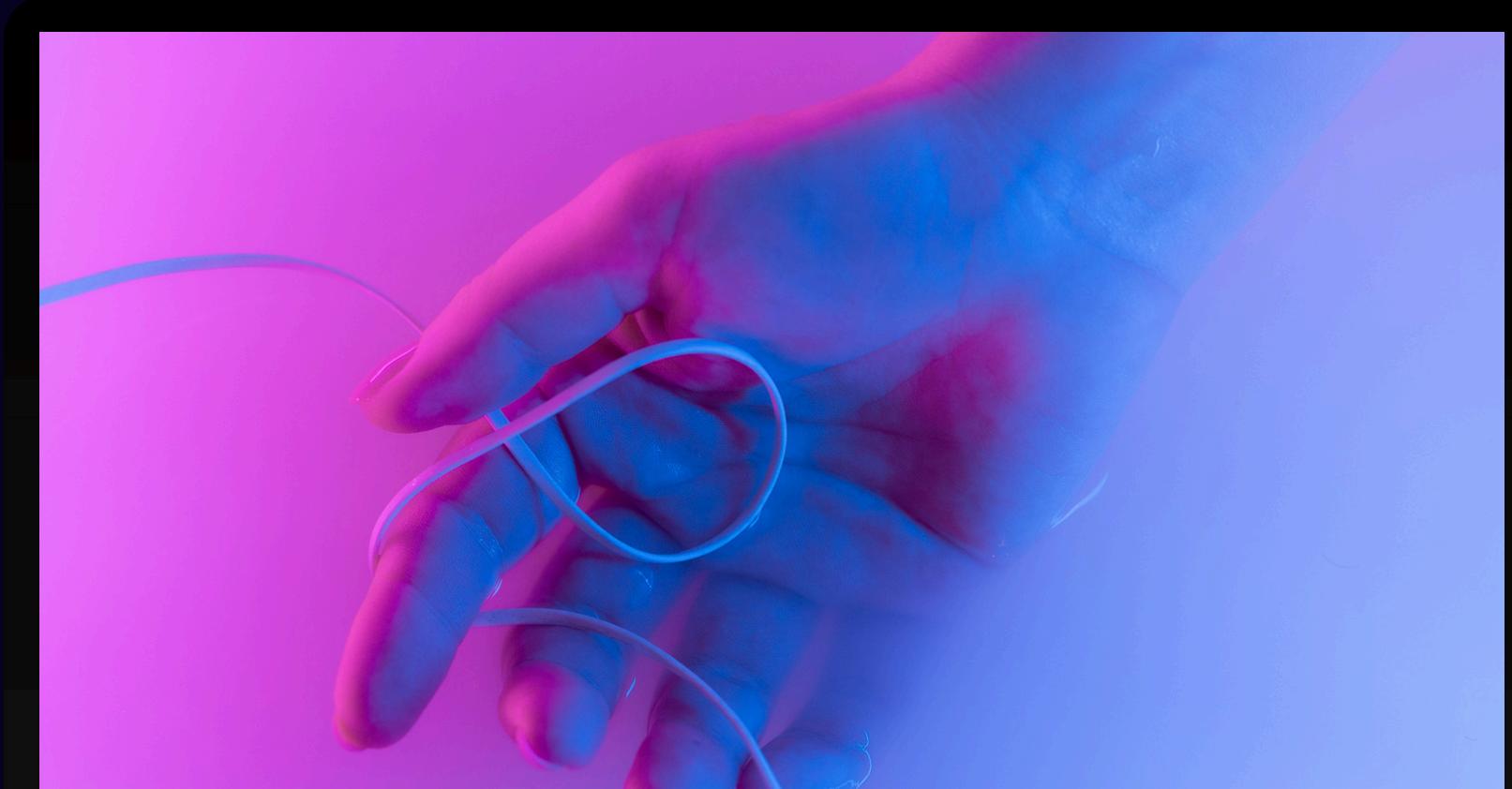
Un trojan è un tipo di malware che si maschera da software legittimo per ingannare gli utenti e infettare i loro sistemi. Questo tipo di malware può consentire agli attaccanti di accedere al sistema infetto, rubare dati sensibili, o installare ulteriori minacce.

Nonostante i motori antivirus abbiano fornito etichette diverse per la minaccia, c'è un accordo unanime sul fatto che il file sia pericoloso e rappresenti una seria minaccia per la sicurezza dei sistemi informatici. Pertanto, è consigliabile eliminare immediatamente il file e adottare misure di sicurezza aggiuntive per prevenire ulteriori infezioni.





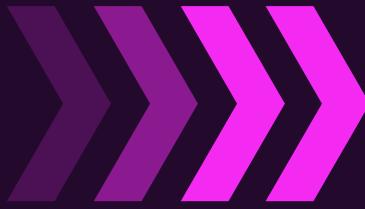
ESERCIZIO SIDE 3



TRACCIA:

Con riferimento alla figura in slide 3, risponde ai seguenti quesiti:

3. Identificare i costrutti noti (creazione dello stack, eventuali cicli)
4. Ipotizzare il comportamento della funzionalità implementata altri costrutti)
5. Fare una tabella per spiegare il significato delle singole righe di codice



COSTRUTTI NOTI





Nel codice assembly visualizzato, possiamo identificare diversi costrutti noti, inclusi quelli relativi alla gestione dello stack, controllo condizionale e chiamate di funzione. Di seguito, una descrizione dettagliata dei costrutti individuati:

La gestione dello stack inizia con l'istruzione “**push ebp**”, che salva il valore del puntatore di base (**EBP**) sullo stack. Successivamente, “**mov ebp, esp**” copia il valore del puntatore dello stack (**ESP**) nel puntatore di base (**EBP**), creando un nuovo frame dello stack. L'istruzione “**sub esp, 4**” alloca spazio sullo stack per una variabile locale.

Le chiamate di funzione sono rappresentate dall'istruzione “**call ds:InternetGetConnectedState**”, che chiama la funzione “**InternetGetConnectedState**” per verificare lo stato della connessione Internet, e dall'istruzione “**call sub_40117F**”, che chiama una funzione, probabilmente per stampare i messaggi di successo o errore.

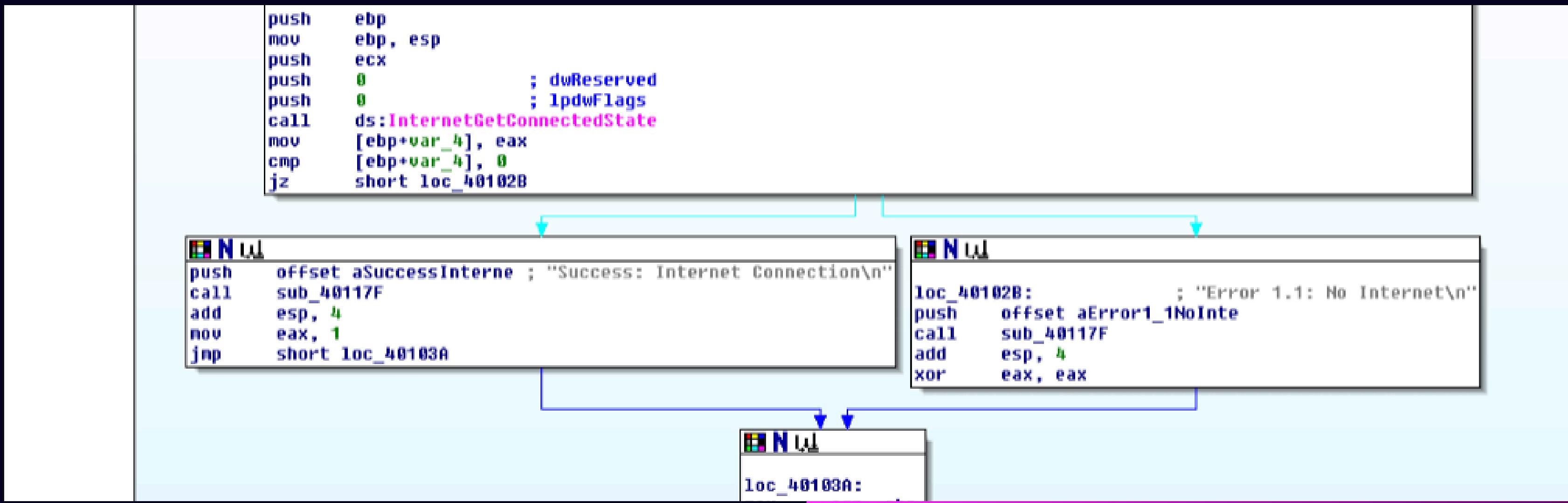
Il controllo condizionale viene gestito con l'istruzione “**cmp [ebp+var_4], 0**”, che confronta il valore della variabile locale “**var_4**” con 0. Se il confronto risulta in zero (cioè, non c'è connessione Internet), l'istruzione “**jz short loc_40102B**” salta all'indirizzo “**loc_40102B**”.

Per quanto riguarda la gestione dei valori di ritorno, l'istruzione “**mov [ebp+var_4], eax**” memorizza il valore di ritorno della funzione “**InternetGetConnectedState**” in “**var_4**”. L'istruzione “**mov eax, 1**” imposta il registro “**eax a 1**”, indicando successo, mentre l'istruzione “**xor eax, eax**” imposta il registro “**eax a 0**”, indicando errore.

I messaggi di output vengono gestiti con l'istruzione “**push offset aSuccessInterne**”, che spinge l'indirizzo del messaggio di successo sullo stack, e con l'istruzione “**push offset aError1_1NoInte**”, che spinge l'indirizzo del messaggio di errore sullo stack.

Infine, la conclusione e il ripristino dello stack vengono effettuati con l'istruzione **mov esp, ebp**, che ripristina il puntatore dello stack al valore del puntatore di base, seguita dall'istruzione **pop ebp**, che ripristina il valore del puntatore di base (**EBP**) dallo stack, e dall'istruzione **retn**, che ritorna alla funzione chiamante.

Non ci sono cicli (loop) esplicativi nel codice visualizzato. La struttura del codice è principalmente sequenziale con salti condizionali basati sul risultato della verifica della connessione Internet.





LA FUNZIONALITÀ



Il codice assembly illustrato nel diagramma mostra una funzione che verifica lo stato della connessione Internet e restituisce un messaggio di successo o errore in base al risultato.

La funzione inizia salvando il contesto corrente dello stack con “**push ebp**” e “**mov ebp, esp**”. Poi passa alcuni parametri (inclusi due valori 0) alla funzione “**InternetGetConnectedState**”, che verifica lo stato della connessione Internet.

Il risultato di questa verifica viene memorizzato in una variabile locale “(**mov [ebp+var_4], eax**)”. Successivamente, il valore di “**var_4**” viene confrontato con 0 (**cmp [ebp+var_4], 0**). Se il risultato è 0 (cioè, non c'è connessione Internet), il flusso di controllo passa a “**loc_40102B**”, che gestisce l'errore.

Se invece c'è una connessione Internet, viene stampato il messaggio "Success: Internet Connection" (“**push offset aSuccessInterne e call sub_40117F**”). L'accumulatore (**eax**) viene impostato a 1 (**mov eax, 1**) per indicare il successo e il flusso passa a “**loc_40103A**”.

In caso di assenza di connessione Internet, viene stampato il messaggio "Error 1.1: No Internet" (**push offset aError1_1NoInte e call sub_40117F**). L'accumulatore (**eax**) viene azzerato (**xor eax, eax**) per indicare un errore. Infine, la funzione ripristina lo stack con “**mov esp, ebp, pop ebp e retn**”, e ritorna al chiamante con il valore di **eax** che indica il risultato: 1 per successo e 0 per errore.

Il codice non contiene cicli esplicativi, ma è strutturato in modo sequenziale con salti condizionali basati sul risultato della verifica della connessione Internet.



TABELLA SPEGAZIONE RIGHE CODICE



Il codice assembly illustrato nel diagramma mostra una funzione che verifica lo stato della connessione Internet e restituisce un messaggio di successo o errore in base al risultato.

La funzione inizia salvando il contesto corrente dello stack con “**push ebp**” e “**mov ebp, esp**”. Poi passa alcuni parametri (inclusi due valori 0) alla funzione “**InternetGetConnectedState**”, che verifica lo stato della connessione Internet.

Il risultato di questa verifica viene memorizzato in una variabile locale “(**mov [ebp+var_4], eax**)”. Successivamente, il valore di “**var_4**” viene confrontato con 0 (**cmp [ebp+var_4], 0**). Se il risultato è 0 (cioè, non c'è connessione Internet), il flusso di controllo passa a “**loc_40102B**”, che gestisce l'errore.

Se invece c'è una connessione Internet, viene stampato il messaggio "Success: Internet Connection" (“**push offset aSuccessInterne e call sub_40117F**”). L'accumulatore (**eax**) viene impostato a 1 (**mov eax, 1**) per indicare il successo e il flusso passa a “**loc_40103A**”.

In caso di assenza di connessione Internet, viene stampato il messaggio "Error 1.1: No Internet" (**push offset aError1_1NoInte e call sub_40117F**). L'accumulatore (**eax**) viene azzerato (**xor eax, eax**) per indicare un errore. Infine, la funzione ripristina lo stack con “**mov esp, ebp, pop ebp e retn**”, e ritorna al chiamante con il valore di **eax** che indica il risultato: 1 per successo e 0 per errore.

Il codice non contiene cicli esplicativi, ma è strutturato in modo sequenziale con salti condizionali basati sul risultato della verifica della connessione Internet.